Attorney Docket No.: 06944.0033
Customer Number: 22,852

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: :

Ashok VADEKAR et al.   :

Serial No.: 09/761,700   :   Group Art Unit: Unknown

Filed: January 18, 2001   :   Examiner: Unknown

For: TIMING ATTACK RESISTANT CRYPTOGRAPHIC SYSTEM

CLAIM FOR PRIORITY

**ATTN: OFFICE OF INITIAL PATENT EXAMINATION'S
CUSTOMER SERVICE CENTER**
Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

Under the provisions of 35 U.S.C. § 119, Applicants hereby claim the

benefit of the filing date of Canadian Patent Application No. 2,243,761, filed July

21, 1998, for the above-identified U.S. patent application.

In support of this claim for priority, enclosed is one certified copy of the

priority application.

Respectfully submitted,
FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, L.L.P.

By: _____
Ernest F. Chapman
Reg. No. 25,961

LAW OFFICES
FINNEGAN, HENDERSON,
FARABOW, GARRETT,
& DUNNER, L.L.P.
1300 I STREET, N. W.
WASHINGTON, DC 20005
202-408-4000

Date: April 10, 2001
EFC/FPD/peg
Enclosure

Office de la propriété intellectuelle du Canada

Un organisme
d'Industrie Canada

Canadian Intellectual Property Office

An Agency of
Industry Canada

Bureau canadien des brevets

Certification

Canadian Patent Office

Certification

La présente atteste que les documents ci-joints, dont la liste figure ci-dessous, sont des copies authentiques des documents déposés au Bureau des brevets.

This is to certify that the documents attached thereto and identified below are true copies of the documents on file in the Patent Office.

Specification and Drawings, as originally filed, with Application for Patent Serial No: **2,243,761**, on July 21, 1998, by **CERTICOM CORP.**, assignee of Ashok Vadekar and Robert J. Lambert, for "Timing Attack Resistant Cryptographic System".

L. Régimbal

Agent certificateur/Certifying Officer

December 27, 2000

Date

Canada

OPIC   CIPO

ABSTRACT

A method for determining a result of a group operation performed an integral number of times on a selected element of the group, the method comprises the steps of :representing

5 the integral number as a binary vector; initializing an intermediate element to the group identity element; selecting successive bits, beginning with a left most bit, of the vector. For each of the selected bits; performing the group operation on the intermediate element to derive a new intermediate element; replacing the intermediate element with the new intermediate element; performing the group operation on the intermediate element and an

10 element, selected from the group consisting of: the group element if the selected bit is a one; and an inverse element of the group element if the selected bit is a zero; replacing the intermediate element with the new intermediate element. In a final step, performing the group operation on the intermediate value and the inverse element if the last selected bit is a zero; and replacing the intermediate element therewith, to obtain the result, whereby each of

15 the bits of the integral is processed with substantially equal operations thereby minimizing timing attacks on the cryptographic system.

TIMING ATTACK RESISTANT CRYPTOGRAPHIC SYSTEM

The present invention relates to the field of cryptographic systems and in particular to a method and apparatus for resisting timing attacks on a cryptographic system.

5

BACKGROUND OF THE INVENTION

Cryptographic systems generally owe their security to the fact that a particular piece of information is kept secret without which it is almost impossible to break the scheme. This secret information must generally be stored within a secure boundary, making it difficult for
10 an attacker to get at it directly. However, various schemes or attacks have been attempted in order to obtain this secret information. One of these is the timing attack.

By way of background current public key cryptographic schemes such as RSA and elliptic curve (EC) operate over mathematical groups F_p^* and $E(Fq)$ respectively. The group operations, called multiplication modulo p, in RSA, and addition of points in EC are repeated
15 in a particular way to perform a scalar operation. In RSA the operand is called an exponent, the operation is called exponentiation and the method of multiplying is commonly known as repeated square-and-multiply. Thus given a number a $\in$ F_p and an integer $0 \le k < p$, the exponent, whose binary representation is k = $\Sigma\ {}^t_{i=o}\ k_i 2^i$ a value a^k mod p may be calculated by repeated use of the square-and-multiply algorithm. Similarly given $g(x) \in F_p m$ and an
20 integer $0 \le k \le p^m - 1$ then $g(x)^k$ mod $f(x)$ may be calculated by this method.

On the other hand, in EC the operand is a scalar multiplier, the operation is called scalar multiplication of a point, and the method is known as double-and-add. Thus if a is a positive integer and P is an elliptic curve point then aP may be obtained by the double-and-add method. Both these methods are well known in the art and will not be discussed further.
25 In RSA, half of all exponentiation operations use a private key. Whereas in EC all scalar multiplications use either a long term private key or a session private key. In each of these cases, the private key is safe due to the difficulty of reversing the exponentiation or multiplication operation as the case may be. This is based on the discrete log problem or the difficulty of integer factorization. As mentioned earlier, an attacker once in possession of
30 the private key (either long term or session) is able to forge signatures and decrypt secret

messages for the attacked entity. Thus it is paramount to maintain the secrecy or integrity of the private key in the system.

Many techniques have been suggested to obtain the private key. The encryption operations are performed either in a special purpose or general purpose processor operating
5   in a cyclic manner. Recent attack methods proposed in open literature have been based on timing analysis of these processors or in other words timing analysis of 'black box' operations. In one instance an attacker by capturing the instantaneous power usage of a processor throughout a private key operation obtains a power signature. The power signature relates to the number of gates operating at each clock cycle. Each fundamental operation as
10   described in the preceding paragraph generates a distinct timing pattern. Other methods exist for obtaining a power signature than instantaneous power usage.

Laborious but careful analysis of an end-to-end waveform can decompose the order of add-and-double or square-and-multiply operations. Either a double or square must occur for each bit of either the exponent or scalar multiplier respectively. Therefore, the places
15   where double waveforms are adjacent each other represent bit positions with zeros and places where there are add patterns indicate bits with ones. Thus these timing measurements can be analyzed to find the entire secret key and thus compromise the system. Thus there is a need for a system which minimizes the risk of a successful timing attack.

20   SUMMARY OF THE INVENTION

This invention thus seeks to provide a cryptographic system where cryptographic operations are performed by a processor in a constant period of time irrespective of the operation being performed whereby a constant amount of time is required for the processing of each bit scalar or a exponent regardless of its value.
25   A method for determining a result of a group operation performed an integral number of times on a selected element of the group, said method comprising the steps of :

    (a) representing said integral number as a binary vector;

    (b) initializing an intermediate element to the group identity element;

    (c) selecting successive bits, beginning with a left most bit, of said vector and for
30   each of said selected bits;

2

(i)   performing said group operation on said intermediate element to derive a new intermediate element;

(ii)   replacing said intermediate element with said new intermediate element;

5   (iii)   performing said group operation on said intermediate element and an element, selected from the group consisting of:

said group element if said selected bit is a one; and

an inverse element of said group element if said selected bit is a zero;

10   (iv)   replacing said intermediate element with said new intermediate element;

(d) performing said group operation on said intermediate value and said inverse element if said last selected bit is a zero; and replacing said intermediate element therewith, to obtain said result, whereby each of the bits of said

15   integral is processed with substantially equal operations thereby minimizing timing attacks on said cryptographic system.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features of the invention will now be described by way of example

20   only with reference the accompanying drawings in which:

Figure 1 is a schematic representation of a data communication system;

Figure 2 is a schematic representation of a binary number recoding scheme;

Figure 3 is a state machine representation for compiling a scalar multiple of a point;

Figure 4 is a pseudocode implementation of the state machine of Figure 3;

25   Figure 5 is a non-state machine pseudocode implementation;

Figures 6 and 7 are respectively a pseudocode and state-machine implementation for a square and multiply scheme; and

Figure 8 is a generalized schematic diagram of an apparatus for implementing the method according to an embodiment of the invention.

30

3

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Referring therefore to Figure 1, a secure data communication system 10 includes a pair of correspondents, designated as a sender A(12), and a recipient B(14), who are connected by a communication channel 16. Each of the correspondents A and B (12,14)

5   includes an encryption unit 18,20 respectively that may process digital information and prepare it for transmission across the channel 16.

Generally, the sender A assembles a data string, which includes amongst others the public key y of the sender, a message m, the sender's short-term public key k and signature S of the sender A. When assembled the data string may be forwarded to the intended recipient

10   B, who then verifies the signature using A's public key. This public key information may be obtained from a certification authority (CA) 24 or sometimes is sent with the message.

For example, in RSA public key encryption, B encrypts a message m for A, which A decrypts. To encrypt the message m, B obtains A's public key (n, e) and represents the message as an integer in the interval [0, m-1]. Next B computes $c = m^e \bmod n$ and sends

15   the cipher text c to A. The entity A then recovers the message m from c by using its private key d to recover $m = c^d \bmod n$. The calculation $c^d \bmod n$ (modulo exponentiation) may be performed by using a well known square and multiply algorithm. Similar computations are required for signing in RSA, and for signing, encryption and decryption in discrete log systems such as ECC.

20   The values used in the computations are expressed as bit vectors, which are then manipulated by the encryption processor in accordance with a particular encryption scheme being used. Thus, referring to figure 2, any bit vector ending in a one can be recoded into an all non-zero vector of +1 and -1 terms. For elements ending in a zero, it is necessary to either maintain the final zero in the recoded form or to perform a further -1 action as a correction

25   after the final zero is processed by the previous rules. As will be apparent later, it is preferable to do the corrective action.

Referring to figure 3 a state machine implementation of an algorithm for computing an integer multiple b of a point P on an elliptic curve is shown generally by numeral 30. In this embodiment, a bit vector b to be processed is fed into the state machine 30. The result of

30   the processing is the value bP where P is a point having coordinates (x,y) on an elliptic curve.

4

The state machine 30 is initialized with a counter i set to the number of bits N in the vector b, and an intermediate value Q stored in a register. Entry into state H0 of the state machine occurs when the first bit (MSB) of b is encountered (not the first non-zero bit). In state H0 the contents of the intermediate Q is doubled and the base point P is added thereto. The bits are sequentially processed causing transitions either back to the current state H0 or to a next state H1. In state H1, Q is doubled and the base point P is subtracted. When the next bit is a zero, the next state is always H1. When the next bit is a one, the next state is always H0. Whenever the current state is H0, an add (+P) occurs, otherwise a subtract (-P) occurs. Once all bits are processed, control exits to the DONE state with the result bP. If the exit condition is encountered while in H0, it is necessary to subtract (-P) a final time from the result.

Implementation of the state machine in pseudocode form requires the availability of a general purpose storage bit for the H state. Figure 4 shows a pseudocode implementation of the state machine where general purpose data and control mechanisms are available. Timing attacks are prevented only if the execution time and power are identical for all possible execution paths through the loop. In this implementation, this requires that the time for the execution path through lines L5, L6 and L7 be the same as that through lines L5, L8 and L9. This implies that the branch instruction (IF) must execute in the same time for false and true conditionals.

Furthermore, the add and subtract operations must take the same time. By adding in the negative value of P, this is more obviously possible. Note that the execution time of all other lines in the algorithm are non-critical to the timing attack resistance, with the exception of lines L14 and L15. These two lines, if executed will increase the total time required, thus revealing that the final H state was zero. This is equivalent to revealing that the final bit of the scalar was a zero. This is unavoidable, but only reveals a single bit of the scalar multiple.

On application specific computing devices, it is likely that there are no general purpose data storage areas nor general purpose assignment and test operators. In this case, the H state control cannot be added in a usual way. Instead, it is necessary to encode the state by branching through distinct code paths.

Referring to Figure 5 shows a 'state-less' pseudocode implementation. Each code execution path corresponds to a distinct state. While this will in general cause code

5

expansion, here there are only two short paths required. Here, as above, timing attacks are prevented only if the execution time and power are identical for all possible execution paths through the loop. In this implementation, this requires that the time for the execution path through lines LL3, LL4 and LL5 be the same as that through lines LL6, LL7 and LL8. In

5   addition, path LL9, LL10 and LL16 must execute in the same time and power as path LL9 and LL11. This will be true in architectures where conditional branch instructions take the same time to branch (to a new location) as to fall through (to the following location). Note that lines LL11, LL12 and LL13 are equal in execution time and power to lines LL16, LL17 and LL18. This is also necessary to prevent timing attacks. Otherwise, H state information

10 would be revealed. As in the previous implementation, a final corrective subtract is required line LL14 when the loop terminates through the H0 path. This again reveals the final H path, or equivalently the final bit of the scalar.

Referring to Figure 6 a pseudocode implementation of a square and multiply operation is shown by numeral 60, while a corresponding state machine implementation is

15 shown in Figure 7. In this implementation, the exponent bit vector b is fed into the state machine 60. The result of the processing is to compute a value M^b. Once again a counter is initialized to the length N of vector b and an accumulator Q is initialized to one. Entry into state H0 of the state machine begins with the MSB of bit vector b. As previously, the bits are sequentially processed causing transitions either back to the current state H0 or to a next state

20 H1. In state H0 the contents of the accumulator Q is squared and multiplied by the base M. In state H1 the accumulator is also squared and then divided by the base M. If the next bit is a zero, the next state is always H1, whereas if the next bit is a one the next state is always H0. Whenever the current state is H0, a multiply occurs otherwise a divide occurs. As previously, once all bits are consumed, controls exits to the DONE state. If the exit condition

25 is encountered while in H0, it is necessary to divide a final line by the base M.

Figure 8 shows a generalized processor implementation in which a state controller 82 is programmed to run the estate code as described earlier. A modulo arithmetic and/or finite field computation unit 84 is provided for computing the square and multiplication and the point additions/subtractions respectively. A counter 86 and a register b is coupled to the state

30 controller for timing the sequential operation of the controller 82.

6

While the invention has been described in connection with a specific embodiment thereof and in a specific use, various modifications thereof will occur to those skilled in the art without departing from the spirit of the invention as set forth in the appended claims.

5   The terms and expressions which have been employed in the specification are used as terms of description and not of limitations, there is no intention in the use of such terms and expressions to exclude any equivalents of the features shown and described or portions thereof, but it is recognized that various modifications are possible within the scope of the claims to the invention.

THE EMBODIMENTS OF THE INVENTION IN WHICH AN EXCLUSIVE PROPERTY OR PRIVILEGE IS CLAIMED ARE DEFINED AS FOLLOWS:

1. A method for determining a result of a group operation performed an integral number
5   of times on a selected element of the group, said method comprising the steps of :

    (a) representing said integral number as a binary vector;

    (b) initializing an intermediate element to the group identity element;

    (c) selecting successive bits, beginning with a left most bit, of said vector and for each of said selected bits;

10   (i) performing said group operation on said intermediate element to derive a new intermediate element;

        (ii) replacing said intermediate element with said new intermediate element;

        (iii) performing said group operation on said intermediate element and
15   an element, selected from the group consisting of:

            said group element if said selected bit is a one; and

            an inverse element of said group element if said selected bit is a zero;

        (iv) replacing said intermediate element with said new intermediate
20   element;

    (d) performing said group operation on said intermediate value and said inverse element if said last selected bit is a zero; and replacing said intermediate element therewith, to obtain said result, whereby each of the bits of said integral is processed with substantially equal operations thereby minimizing
25   timing attacks on said cryptographic system.

2. A method as defined in claim 1, said group being a multiplicative group F_p^* said group element being an integer, and said group operation being exponentiation g^a and an
30   inverse element being the multiplicative inverse $1/g$.

3. A method as defined in claim 1, said group being an additive group E(F_{2^μ}) and said group operation being addition of points.

5 4. A method as defined in claim 1, said group being an additive group E(F_q), said group element being a point P with coordinates (x,y) on the elliptic curve, and said group operation being the scalar multiple kP of said point and an inverse element being the negative –P of said point.

10 5. A method as defined in claim 1, said integral value being a private key k.

6. A method of performing a selected group operation on a scalar and a selected element of said group, in a cryptographic processor, said method comprising the steps of :

representing said scalar as a binary vector;

15 recoding said binary vector to produce a signed digit representation of plus one and minus one digits;

selecting each of said recoded bits sequentially and for each of said selected bits performing said group operation on an intermediate element to derive a new intermediate element; and adding or subtracting said selected element to said

20 intermediate element in accordance with said sign if said digit being selected; and

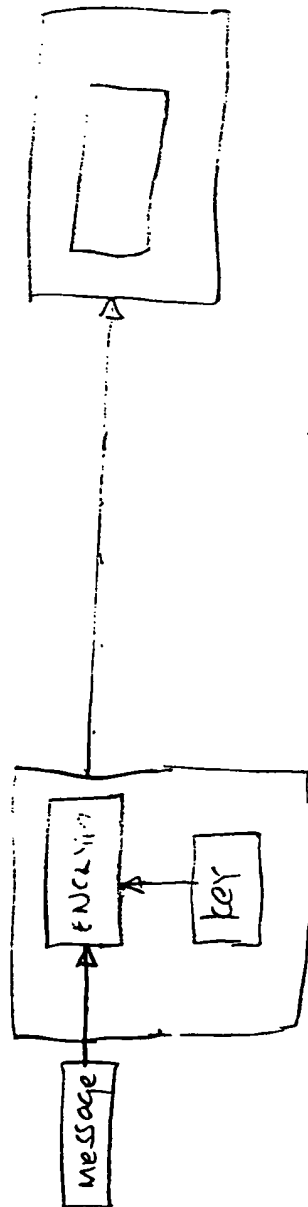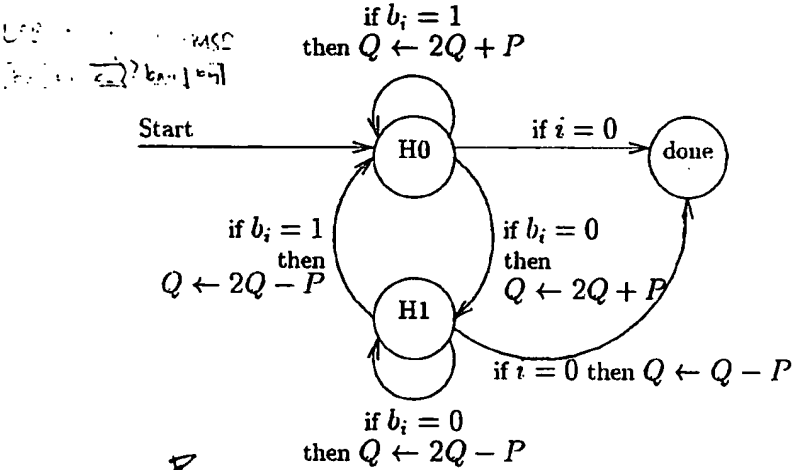outputting said intermediate value as a result of said group operation.

Figure 1

Figure 2

$$\text{if } b_i = 1$$
$$\text{then } Q \leftarrow 2Q + P$$

Start

H0

if $i = 0$

done

if $b_i = 1$
then
$Q \leftarrow 2Q - P$

if $b_i = 0$
then
$Q \leftarrow 2Q + P$

H1

if $i = 0$ then $Q \leftarrow Q - P$

if $b_i = 0$
then $Q \leftarrow 2Q - P$

State	Input	Next	Action
H0	$i < 0$	done	subtract
H0	$b_i = 0$	H1	double, add
H0	$b_i = 1$	H0	double, add
H1	$i < 0$	done	
H1	$b_i = 0$	H1	double, subtract
H1	$b_i = 1$	H0	double, subtract

Figure 3

```
begin:
     i := N              ; Start from MSB                        L1
     Q := 0              ; Initialize accumulator                L2
     H := 0              ; Initialize state                      L3

loop:                    ; For all bits
     Q := Q + Q          ; Double accumulator                    L4

     if H=0              ; If H state is set                     L5
         Q := Q + P      ;     Add base point to accumulator     L6
         goto endLoop    ;                                       L7
     else                ; Else
         Q := Q + (-P) ; ;     Subtract base point               L8
         goto endLoop    ;                                       L9

endLoop:
     H := b[i]           ; set H state to value of b[i]          L10
     i := i - 1          ; Process next bit                      L11
     if i>0              ; If bit exists                         L12
         goto loop       ;     continue at top of loop           L13

     if H=0              ; If exiting from H=0 state             L14
         Q := Q + (-P) ; ;     correct result by final subtract  L15
     end                                                         L16
```

Figure 4

```
begin:
    i := N           ; Start from MSB                          LL1
    Q := 0           ; Initialize accumulator                 LL2

H0:                  ; State entry point
    Q := Q + Q       ; Double accumulator                     LL3
    Q := Q + P       ; Add base point to accumulator          LL4
    goto endLoop     ; Branch to end of loop tests            LL5

H1:                  ; State entry point
    Q := Q + Q       ; Double accumulator                     LL6
    Q := Q + (-P)    ; Subtract base point from accumulator   LL7
    goto endLoop     ; Branch to end of loop tests            LL8

endLoop:             ; End of loop tests
    if b[i]=1        ; If current bit is set                  LL9
        goto nextH0  ;     follow H0 path                     LL10
                     ; Else fall into H1 path

nextH1:              ; H1 path
    i := i - 1       ; Process next bit                       LL11
    if i>0           ; If bit exists                          LL12
        goto H1      ;     execute H1 state                   LL13
    Q := Q + (-P)    ; Else correct result and end            LL14
    end                                                       LL15

nextH0:              ; H0 path
    i := i - 1       ; Process next bit                       LL16
    if i>0           ; If bit exists                          LL17
        goto H0      ;     execute H0 state                   LL18
    end              ; Else end                               LL19
```

Figure · 5

begin:

$i := N$

$Q := 1$

H0: $Q := Q \cdot Q$ (Q^2)

$Q := Q \cdot m$

Goto endloop

H1: $Q := Q \cdot Q$

$Q := Q/m$ $(a \cdot m^{-1})$

endloop: if $b[i] = 1$ goto next H0

next H1: $i = i - 1$

if $i > 0$ goto H1

$Q = Q/m$

end

next H0: $i = i - 1$

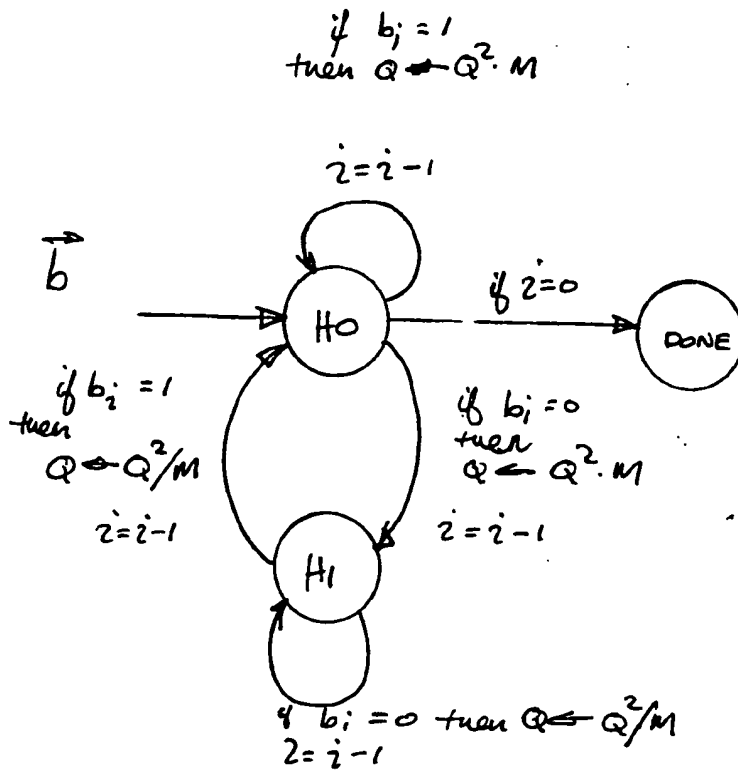if $i > 0$ goto H0

end

60

Figure 6

if $b_i = 1$
then $Q \leftarrow Q^2 \cdot M$

$i = i - 1$

$\vec{b}$

H0

if $i = 0$

DONE

if $b_i = 1$
then
$Q \leftarrow Q^2/M$

$i = i - 1$

if $b_i = 0$
then
$Q \leftarrow Q^2 \cdot M$

$i = i - 1$

H1

if $b_i = 0$ then $Q \leftarrow Q^2/M$
$i = i - 1$

Figure 7

P or Q_x

b

Intermediate q

Q

b_i

i

COUNT   86

FSM

STATE CONTROLLER

F_P / F_{2^m}

CLK

84   FIGURE 8   82